Hindawi
Journal of Sensors
Volume 2017, Article ID 8241910, 15 pages
https://doi.org/10.1155/2017/8241910

Research Article

Three Ways to Improve the Performance of Real-Life Camera-Based Fall Detection Systems

Glen Debard,[1,2] Marc Mertens,[1,3] Toon Goedemé,[2,4] Tinne Tuytelaars,[2,5] and Bart Vanrumste[6,7,8]

[1] *Thomas More Kempen, MOBILAB, Kleinhoefstraat 4, 2240 Geel, Belgium*
[2] *ESAT-PSI, KU Leuven, Leuven, Belgium*
[3] *DTAI, Department of Computer Science, KU Leuven, Leuven, Belgium*
[4] *EAVISE, KU Leuven Campus De Nayer, Sint-Katelijne-Waver, Belgium*
[5] *iMinds, Leuven, Belgium*
[6] *eMedia, Group T, KU Leuven Technology Campus, Leuven, Belgium*
[7] *ESAT-STADIUS, KU Leuven, Leuven, Belgium*
[8] *IMEC, Leuven, Belgium*

Correspondence should be addressed to Glen Debard; glen.debard@thomasmore.be

Received 14 June 2017; Accepted 12 September 2017; Published 23 October 2017

Academic Editor: Stefano Stassi

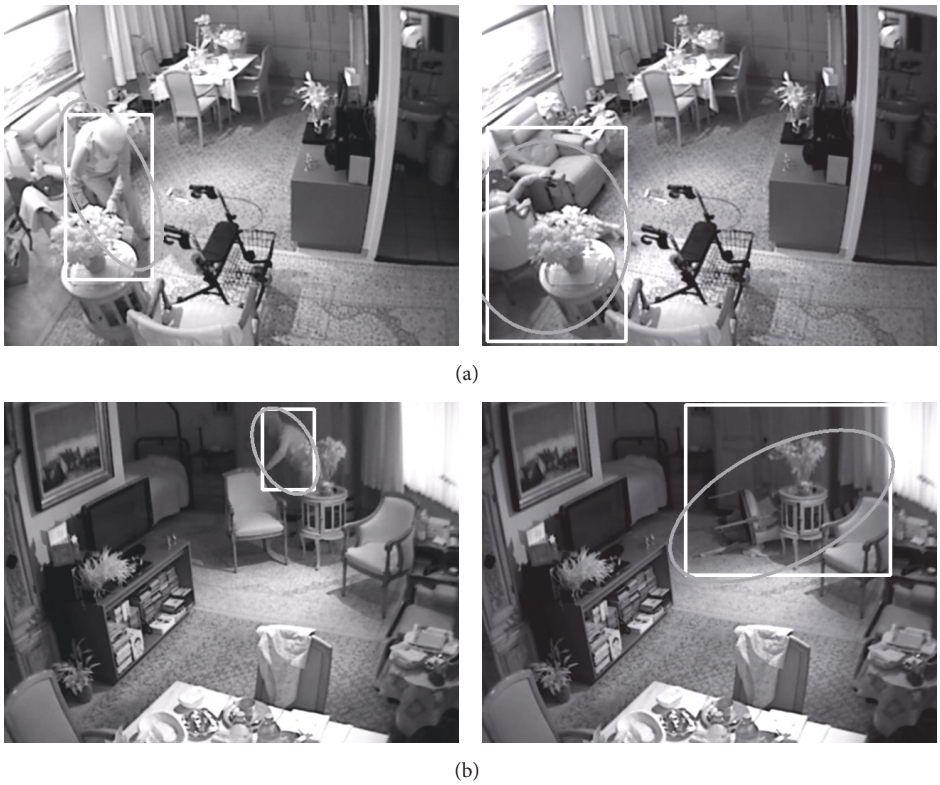
More than thirty percent of persons over 65 years fall at least once a year and are often not able to get up again. Camera-based fall detection systems can help by triggering an alarm when falls occur. Previously we showed that real-life data poses significant challenges, resulting in high false alarm rates. Here, we show three ways to tackle this. First, using a particle filter combined with a person detector increases the robustness of our foreground segmentation, reducing the number of false alarms by 50%. Second, selecting only nonoccluded falls for training further decreases the false alarm rate on average from 31.4 to 26 falls per day. But, most importantly, this improvement is also shown by the doubling of the AUC of the precision-recall curve compared to using all falls. Third, personalizing the detector by adding several days containing only normal activities, no fall incidents, of the monitored person to the training data further increases the robustness of our fall detection system. In one case, this reduced the number of false alarms by a factor of 7 while in another one the sensitivity increased by 17% for an increase of the false alarms of 11%.

1. Introduction

The group of persons aged 65 years and over are affected the most by falls and their negative health consequences. Thirty to forty-five percent of this group that live at home fall at least once a year [1], making fall incidents a major cause of health related problems for older persons. There can be both physical complications such as dehydration, pressure ulcers, and even death as well as psychological consequences such as fear of falling, loss of self-confidence, and loss of independence [1, 2]. One determining factor that influences the severity of the consequences of the fall is the amount of time spent on the floor [2]. Fall detection systems that can help ensure timely aid are therefore needed.

The Personal Emergency Response System (PERS) [3] is a commercially available solution for this. But the fallen person has to press the button manually. Even when he or she is not unconscious after a fall and can reach the button, 80% does not use it to call for help [2]. An automatic fall detection system which does not need user intervention can overcome this problem. Because of the importance of this issue, a lot of research has been done to solve the fall detection challenge, as can be seen in the numerous available review articles [4–19]. There are different ways to detect a fall; Mubashir et al. categorizes them, for example, in three categories: wearable sensors, vision, and ambient/fusion [14].

However, each system has its drawbacks. For example, a wearable system has to be worn at all times. This can be



(a)

(b)

Figure 1: Some examples of falls that end occluded and are therefore excluded from *DS_restricted* (see Section 3.1). The white box represents the detected bounding box (see Section 3.3). The gray ellipse represents the bounding ellipse (see Section 3.3). (a) The person falls backwards behind a table and also pushes a chair away. (b) The person falls backwards behind a table and also the chair, which she was holding, falls over.

obtrusive for the monitored older person. Also, during the night or while taking a bath or shower, the system is often not worn [20]. Contactless methods, such as the camera-based system used in this research, can overcome this limitation.

A lot of research has been done, but most of the proposed techniques do not cope well when used in real-life situations. For example, many camera-based fall detection algorithms use background subtraction to find the person in the image. Most researchers [21–26] assume that the silhouette that results from these background subtraction techniques is an accurate representation of the person. However, this is not always the case. Other objects and persons, occlusions, and changing illumination conditions often interfere with the segmentation [27]. These conditions are underrepresented in most simulation data sets (DS). As almost all published research is only validated using simulated data [16, 28], often these challenges remain untested. In our previous work [28], we showed that our fall detection algorithm that was based on a simple background subtraction method performed similarly as the state-of-the-art on a publicly available simulation data set. However, it generated a large amount of false alarms when tested on a real-life data set. Thus, it is important to increase the robustness of the used algorithms. For example, Belshaw et al. [22] use optical flow to increase their robustness. Our approach uses a particle filter in combination with a person detector for this purpose. Earlier in [29], we tested our new approach on a newly created publicly available simulation data set that is based on our real-life data set [30] and got promising results. As first contribution, we show the results of this new approach on our real-life data set.

In previous work [28], we saw that in real-life often the person is occluded, e.g., behind a table, door, and so forth, while falling. Some examples are shown in Figure 1. Normally we try to train our systems using all available data, also including occluded falls in the training set. In previous work, always a high number of false alarms were generated. One of the causes could be that the fall detector is also trying to detect these kinds of falls, often not successfully. The extracted feature values can differ in a high degree with the ones that represent a visible fall but look similar as a normal activity. As a second contribution, we show that training a fall detector using only nonoccluded falls can increase its performance.

Most fall detectors are based on a generic model that is trained using fall data and activities of daily life (ADL) of one or more participants. However, every person has his own way of living. The executed activities differ from person to person and also their walking pattern is unique. We propose incorporating online training to take these differences into account. At installation time the system is based on a generic model. During the next days, the system constantly monitors the person and adapts the model to take this person's specific

life pattern into account. In wearable approaches, this is more commonly used. Medrano et al. showed that personalization increases the performance of smartphone-based fall detectors [31]. They use different recorded falls from other persons to build a model. This is then personalized using the ADL of the monitored person. This type of personalization is not yet common in computer vision based fall detection algorithms. One example is from Yu et al. [26]. They use a single camera and an online one-class support vector machine to create a person-specific fall detection system. They only use ADL from different persons to create a generic model that is able to detect outliers. While using the system, the model is constantly retrained using the ADL of the monitored person to get a more person-specific model. To decrease false alarms, they include two rules to determine a fall: a fall is only reported when a large movement is detected by using the motion energy image and the person should remain on the floor longer than a certain time interval. We on the other hand use a different discriminative approach. As a final contribution, we show that using a generic model that is trained using both falls and ADL from different persons and retrain it afterwards by adding only negative data containing ADL of the monitored person can improve the performance of a fall detector. Below, we show how our approach is built and how it performs.

In this paper we first show some related work before elaborating our approach more in depth. We discuss the used real-life data set to show the performance of our fall detector in Section 3.1. Also the more restricted data set containing only nonoccluded falls and the additional videos for online training are discussed in this section. In Section 3.2, we explain how we combined our background subtraction method with a particle filter and a person detector. The extracted features that are used as input for our classifier are described in Section 3.3. After this, we explain how our approach for a personalized detector is implemented in Section 3.5. Next, we show the results of our three different contributions in Section 4. We discuss these results and propose some future paths to solve the fall detection challenge in Section 5. We conclude the paper in Section 6.

2. Related Work

As stated above, camera-based fall detection systems have the advantage that they are contactless. Because of this, several research groups have focused on their development and explored a whole range of different approaches. Often a single camera is used to try to detect a fall in a room [21–26]. Regular RGB cameras have the advantage that the recorded images can be used to learn the root cause from this fall afterwards. For example, during the monitoring period, we discovered that one of the participants fell twice because she wanted to take something from the bottom shelf of her cupboard. Rearranging the cupboard prevented further fall incidents. Cameras can also be combined in a multicamera network using early or late fusion [32, 33]. With early fusion it is possible to first extract a 3D figure of the person to try to make the fall detection more robust [32]. The commercialization of the Microsoft Kinect also provides the possibility of directly extracting 3D images in an affordable way. This has led to the usage of this sensor to detect falls [34, 35]. Another path that is sometimes used is thermal imagers [36]. These outperform normal cameras in dusky environments. Also an RGB camera with an appropriate filter can be used together with a near-infrared light source as a cheap alternative in these circumstances.

At the algorithmic level, there are two different approaches that are often used to detect falls: those that try to detect unusual events [23, 32, 37, 38] and those that try to detect the action of falling directly [21, 24–26, 39]. The former use indirect evidence, such as prolonged inactivity at unusual locations. The latter extract features of the movement of the person or changes in their posture to try to detect the fall. For this, background subtraction is often used to find the moving foreground objects [21–26], the biggest of which is mostly assumed to be the person. Domain knowledge is then often used to implement simple yet robust fall features, such as the aspect ratio of the bounding box around the person [24–26, 39], or the angle of the surrounding ellipse [24–26, 39]. Another commonly used technique is motion histograms [24, 25, 40].

3. Methods

To detect falls, the region where the person is present in the image has to be detected. From this region, some features can be calculated to classify the fall. Section 3.2 explains our approach to detect the elliptical region of interest (ROI) in the image. The features that are extracted from this and how they are used to classify a fall are elaborated on in Section 3.3. Next, our approach to test the effects of online training is discussed in Section 3.5. But first we start with Section 3.1, in which an overview of the different real-life data sets used for both training and validation of our experiments is given.

3.1. Data Set (DS). During the last years, seven older persons with a high risk of falling were monitored at their place of residence continuously during a period of three months up to two years. This represents an extensive real-life data set consisting of 29 falls and a huge amount of other activities of daily life. All videos are processed using greyscale values. More information about the data set itself and an overview of the different falls can be found in [28].

For our experiments, we used three different combinations of videos from this real-life data set. Table 1 gives an overview per person of the amount of falls and hours of data included in each data set. Because we only have a small number of falls, we do not divide the data in fixed training and test sets. We use tenfold cross-validation in which each video fragment is kept as a whole.

3.1.1. DS_complete. Only part of the available falls and other videos was used in data set *DS_complete* for our experiments. Two inclusion criteria were formed for the fall to be included in these tests. These are the same as used in previous research [28]. The first criterion was that only the person falling should be visible. This is because the extraction of the fall features can

Table 1: Overview of number of falls and number of hours of video per person included in each data set. The data set *DS_complete* contains all falls that meet our inclusion criteria. Data set *DS_restricted* is a subset of *DS_complete* with the additional restrictions that the fall should not be occluded and the person should stay on the floor for over 30 seconds after falling. The data set *DS_personalized* contains additional videos with only nonfall data.

Participant	*DS_complete*		*DS_restricted*		*DS_personalized*
	Falls	Video data	Falls	Video data	Video data
A	12	244.7 h	7	124.7 h	0 h
B	1	24 h	1	24 h	0 h
C	6	144 h	2	48 h	120 h
D	1	24 h	1	24 h	120 h
E	2	48 h	0	0 h	0 h
Total	22	532.7 h	11	220.7 h	240 h

fail if another person is present during the fall. The presence of another person in a different part of the fragment or in another room of the house was allowed. The second criterion was that the fall should happen within the camera's field of view; otherwise it can not be detected. Twenty-two falls met these inclusion criteria. If available, for each fall a fragment of 24 hours was used in which the fall happened in the last 20 minutes. In two cases for participant A, the 24 hours of video was not available due to prior hard disk crashes. In these cases, a fragment of 140 minutes was used instead. For the other twenty falls the full fragment of 24 hours was used.

3.1.2. DS_restricted. During our research, we saw that a high number of falls were occluded; some examples are shown in Figure 1. Because of this, also a more restricted data set *DS_restricted* was used in which the inclusion criteria were extended with two extra restrictions: first, the fall should not be occluded; second, the person should remain on the floor for longer than 30 seconds after falling. The latter restriction is added since these falls are the most important to detect. If the person is able to get up unaided, it is very probable that he or she does not require help. And if help is still needed, the person can warn someone him/herself. Eleven falls met these criteria: two with a fragment length of 140 minutes and nine with the full 24 hours of data. Person E (see Table 1) got up without aid after remaining on the floor for only a couple of seconds.

3.1.3. DS_personalized. To test our approach for a personalized training system, additional videos were used for two persons. We did not use the other persons because, to train and validate a model, sufficient fall data is needed. A huge amount of data is available in the real-life data set, but as a first test we decided to use up to five additional days of video for each of these two persons. These videos only contained negative examples or activities of daily life, no positive data or falls. The data set *DS_personalized* is a combination of five additional videos of 24 hours for both persons.

3.1.4. Experiments. We ran a couple of different experiments. For the first experiment, we used the data set *DS_complete*. The training was executed using the complete data set using tenfold cross-validation. The video segments are added as a whole to either the test or the training set. The second experiment was to show the effect of using only good data for the training of the fall detector; we again used tenfold cross-validation to classify the videos of *DS_restricted*. To compare the results with the ones obtained using *DS_complete*, we trained a fall detector using all videos from *DS_restricted* and using this trained model we classified all videos from *DS_complete* that were not included in *DS_restricted*. The third experiment is to show the effect of personalization or online training. Since several participants were included in our data set, we can simulate an installed system by using a base training set that is created by removing one of these persons from *DS_restricted*. To generalize the results, a different training set is created for each available combination of days of this person from *DS_personalized* combined with the base training set. As mentioned above, these additional videos only contained normal ADL, no falls. Using these different training sets, we can show the effect of adding one to five days of personalized training data.

3.2. Foreground Detection. Previously, our foreground detection was based on background subtraction (BGS) using an approximate median (APM) filter [41]. Shadows were removed using cross correlation. The foreground was further cleaned up using an erosion/dilation step on all foreground pixels. This is a relatively standard workflow for BGS. A more detailed explanation can be found in [28]. One of the conclusions of this work was that a more robust foreground segmentation could improve the performance of our fall detector by reducing the amount of generated false alarms. However, exploratory tests [28] using two background subtraction algorithms available in OpenCV [42], an improved and adaptive mixture of a Gaussian model [43] and a probabilistic method that uses Bayesian inference [44], produced only minor improvements mainly because we use gray-level images instead of color images. Here, we explore the use of a tracker to increase the robustness of our foreground detection. The general idea is to follow a person in the foreground image using an ellipse that surrounds the foreground blob that corresponds to this person. We chose a particle filter (PF) [45] because it is able to cope with nonlinear motions and we combined it with a people detector [46]. A PF considers multiple state hypotheses simultaneously so it can deal with

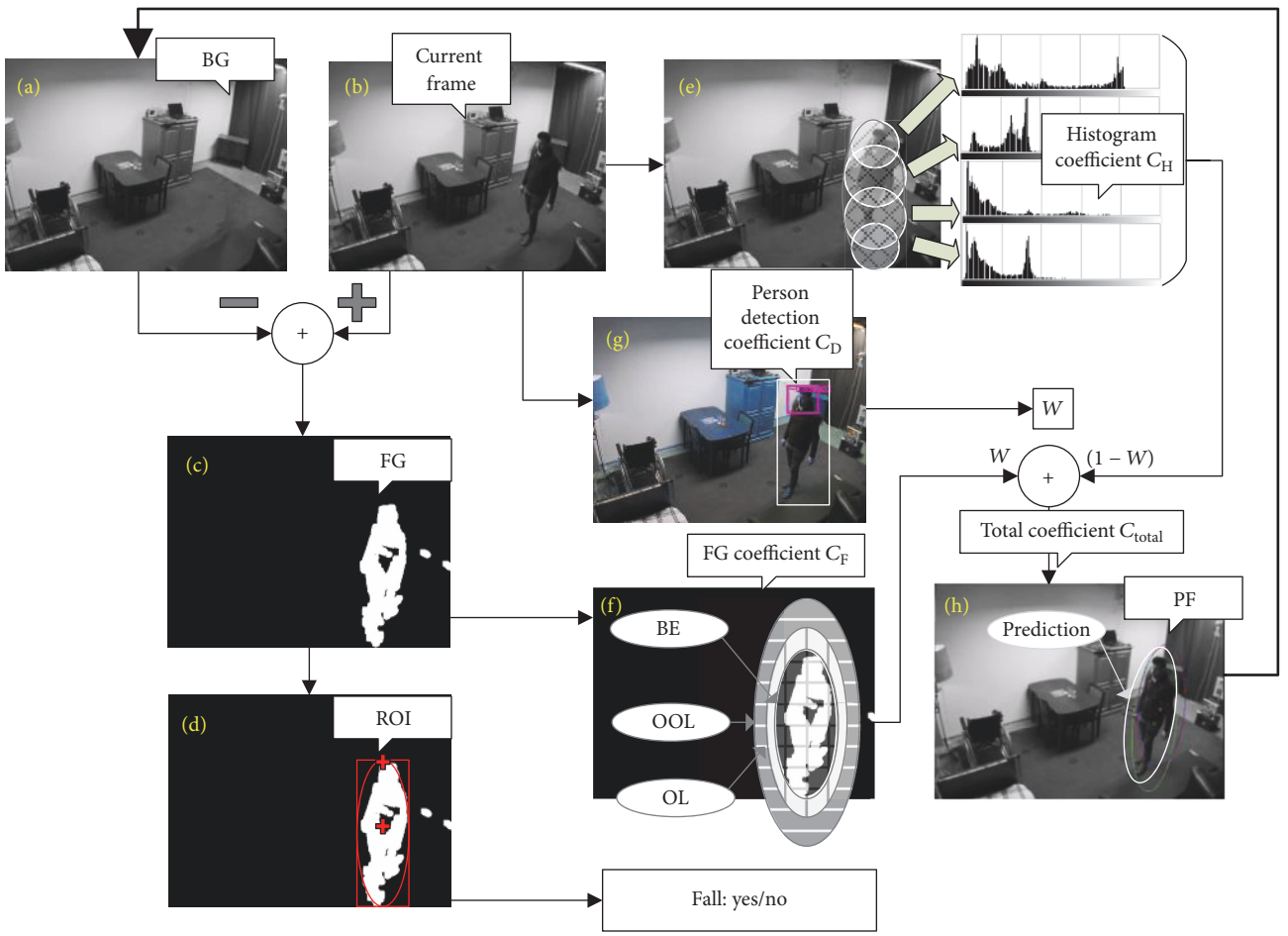


Figure 2: Overview of implementation of foreground (FG) segmentation based on robust background (BG) subtraction using a particle filter (PF) with three different measurement coefficients: FG, histogram and person detection coefficient. (a) BG model. (b) Current input frame. (c) Detected FG. (d) Determined region of interest (ROI) as biggest foreground object (crosses indicate center and top of bounding ellipse (BE)). (e) Histogram coefficient used by PF. (f) FG coefficient used by PF (OL: first outer layer; OOL: second outer layer). (g) Person detection coefficient that controls weight-factor w (small rectangle represents detected upper body; large rectangle represents extrapolated person). (h) Prediction of PF used to update BG model selectively slow inside prediction, fast outside.

short-lived occlusions and can recover when it loses track for a short time.

Our experiments showed that using a PF using only foreground segmentation or a normal histogram did not work well with our challenging real-life data. One reason for this is the lack of color information. Therefore, our implementation uses a combination of foreground segmentation, a weighted structural intensity histogram, and an upper-body detector to determine the weights of the different particles, the combination of which provides the possibility of following the person in the image. Each part has its own function. The upper-body detector helps to prevent and solve tracking losses. The foreground segmentation performs best for tracking the person while he is moving. The weighted structural intensity histogram and the upper-body detector help the tracker during periods with low motion when the foreground might be integrated in the background. It also helps with reducing the effects of ghost formation when the person has been integrated in the background. In this case, the intensity histogram is more robust than the upper-body detector given the low recall of the detector.

To make the background update more robust, the prediction of the PF was used as feedback. The BG was updated more slowly inside of the prediction of the PF and faster outside of this region. This reduced the appearance of a ghost figure while other changes in the image (e.g., changes in lighting, other moving objects) were integrated faster. An overview of the implementation of the foreground detection is shown in Figure 2. A more in-depth explanation of the implementation of the PF can be found in the Appendix. The different configuration values were optimized using visual inspection on a set of validation videos containing only nonfall data and were kept fixed unless specified otherwise.

There are different ways to determine the region of interest (ROI) from which the fall features can be extracted. One possibility is using the predicted ellipse of the particle filter. The performance of this predicted ellipse was benchmarked in a multicamera setting in [47]. It outperformed

Figure 3: Extraction of salient subject features used for fall detection. Purple rectangle: bounding box of subject; white fine segmented line: best-fitting ellipse within subject bounding box; double green diamond: center of mass; blue octagon: head position (small, filled black rectangle is not part of feature extraction set; it is inserted in the figure presentation here to maintain subject anonymity) [28].

multiple dedicated multicamera trackers. This showed that the prediction of the tracker is good for following persons performing normal activities of daily life (ADL). However, a previous study using a challenging simulation data set [29] showed that the fall detector performed best when using the biggest foreground object in the image as ROI and not when using the prediction of the particle filter. This could be explained by the fact that a tracker smooths out sudden abrupt movements of which a fall is a good example. Increasing the reaction speed of the tracker would cause it to be able to better follow the falls, but it would also cause the tracker to be less robust and lose track more often. This could be counteracted by increasing the number of particles, but then also the processing time would increase a lot since the coefficients described above have to be calculated for each particle. The fall detection features are extracted from the biggest detected blob available in the foreground image (see Figure 2(d)). This relies on the assumption that only one person is visible in the image. In real situations, sometimes this is violated. A solution for this is to detect if multiple persons are present and if so, deactivate the system. In most cases, this can be done because the other person could call for help. But, in case of a married couple where the person that is taking care of his/her spouse with cognitive impairment falls, this can cause severe problems. In this case, it is better to make the tracker itself work for multiple persons as done by Young-Sook and HoonJae [48]. For the moment, this is not implemented yet.

3.3. Fall Detection Features. There are numerous ways to detect a fall. Our fall detector used the five features that are most widely used in the literature: aspect ratio (AR) [24–26, 39], change in AR (CAR), fall angle (FA) [24–26, 39], center speed (CS) [49], and head speed (HS) [24] (see Figure 3). More detailed information on how these features are calculated can be found in [28].

When looking to a fall in more detail, four phases can be distinguished as stated by Noury et al. [50]: the prefall, critical, postfall, and recovery phase. All four of these phases contain valuable information to detect a fall. Our approach to include this temporal information was to use a feature vector with a stride length of one second that contains the mean and the maximum values of these features calculated over different time slots from before, during, and after the fall. These different time slots are shown in detail in Figure 4. We have one such feature vector for each time slot of one second covering a certain time frame.

3.4. Classifier. These feature vectors can then be used by a support vector machine (SVM) to determine if this feature vector corresponds to a fall or a nonfall. For this, first the SVM has to learn a model based on a training set containing both fall and nonfall examples. Since only a small number of fall and a huge number of nonfall feature vectors were available, a linear SVM was used to reduce the problem of overfitting and increase the processing speed. To prevent all vectors from being classified as nonfall, different weights were used for negative (w_C) and positive data ($1 - w_C$). The F_β-measure (see below) was used as cost-function for finding the best combination of the weight w_C and the regularization parameter of the SVM. This way, the importance of the sensitivity (SENS) and positive predictive value (PPV) could be weighted appropriately.

$$\begin{aligned} \text{SENS} &= \frac{\text{TP}}{(\text{TP} + \text{FN})} \\ \text{PPV} &= \frac{\text{TP}}{(\text{TP} + \text{FP})} \\ F_\beta &= \left(1 + \beta^2\right) \frac{\text{PPV} \cdot \text{SENS}}{\beta^2 \cdot \text{PPV} + \text{SENS}}, \end{aligned} \tag{1}$$

with TP being the amount of true positives, FN the amount of false negatives, and FP the amount of false positives. Tenfold cross-validation over the complete data set was used for the evaluation of the fall detection algorithm. To reduce the false

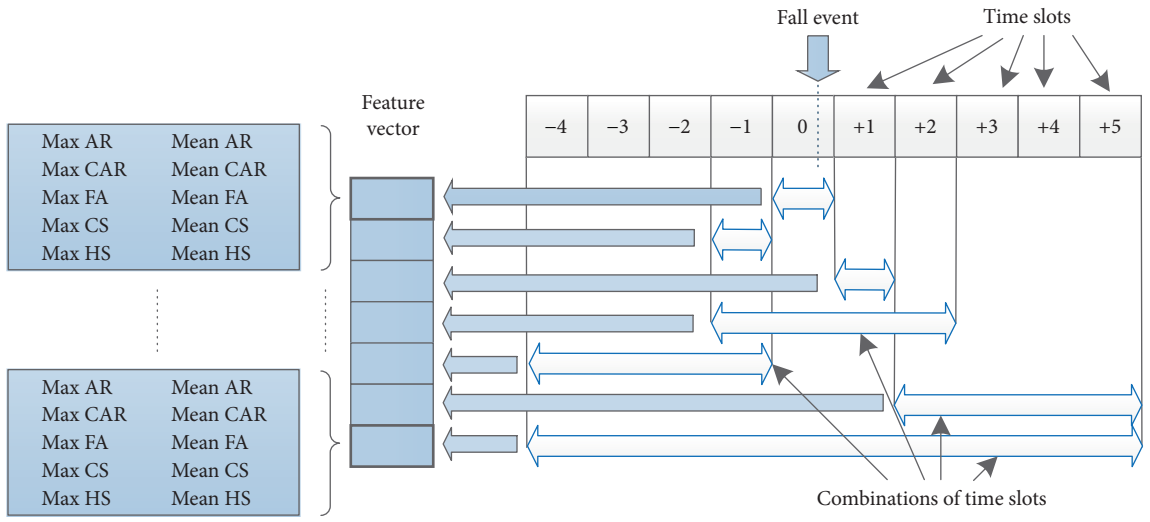


FIGURE 4: Overview of the contents of the feature vector (FV) used by the support vector machine (SVM; Section 3.3). The complete video is split into discrete, one-second time slots. One FV is created for each time slot. A FV contains information about the current time slot and (combinations of) other time slots. Each FV part consists of 10 features as shown (AR = aspect ratio, CAR = change of AR, FA = fall angle, CS = center speed, and HS = head speed) [28].

alarm rate, after each time slot was classified by the SVM, a median filter with length of three was executed over all time slots to remove single detection. Additionally, bursts of detection were grouped using nonmaximum suppression.

3.5. Personalized Detector. Most fall detection algorithms use a generic model based on training data recorded using one or more participants. However, it is very difficult to take into account the huge range of activities of daily life that a person can execute. Another challenge is that every person is unique, and their movement pattern can differ from other persons. Also the activities that are executed differ from person to person. In an ideal world, a fall detection system should use fall and nonfall information from the person that it is monitoring. Since falls are rare, capturing fall data from the person that is being monitored is difficult and could take a long time. Waiting for this to start to be able to detect future falls is not feasible. A better way is to install a fall detection system that is based on generic information and then use the negative, nonfall instances that are being captured to increase the robustness of the system. This way, the living pattern of this person could be learned and taken into account to decrease the number of false alarms.

Our approach for online training is to install the system and then retrain the model every 24 hours. This is easier and faster to implement then adapting the model every time an activity is captured. To show which performance gain can result from this, we use our existing real-life data set. As discussed in Section 3.1, we created a different training set for each available combination of days from the person from *DS_personalized* combined with the base training set. As mentioned above, these additional videos only contained normal ADL, no falls. A model was trained using each of these new training sets. The regularization parameter of the SVM is kept fixed, but the weight of the nonfall data w_{OT} has to be decreased with the quantitative growth of the training set. If w_{OT} would not be decreased, the weight of the nonfall data would become too high, and the sensitivity of the system would decrease. w_{OT} is calculated as follows:

$$w_{\mathrm{OT}} = \frac{w_C}{\left(1 + Z\left(N_{\mathrm{OT}}/N_{\mathrm{orig}} - 1\right)\right)} \tag{2}$$

Z is a factor that can be changed online to regulate the effect of the online training. N_{OT} is the number of feature vectors in the complete training set for online training, while N_{orig} is the number of feature vectors in the original base training set. To test the performance of these personalized detectors, we used all fall videos from the selected person contained in *DS_complete*.

4. Results

4.1. Robust Foreground Detection Validated Using DS_complete. The results of the fall detector based on our robust background subtraction method is shown in Table 2 for $\beta = 10$ for the F_β-measure. The results of a previous study using our less robust foreground detection method from [28] are shown for reference. These earlier experiments were executed with a data set that is very similar to data set *DS_complete*. One fall of participant A was not included before and one of her falls used to have 24 hours of video instead of the now available 140 minutes. In that study we also showed that our original method performed similarly as the state of the art on a publicly available simulation data set.

If we compare the results of our robust BGS algorithm with these of our original BGS [28], there is a large decrease in false alarms for a similar sensitivity. The original system generated 1360 false alarms for a sensitivity of 38.1%, while our more robust system almost halves the amount of false alarms while detecting more falls. The results of the fall

Table 2: Results of robust background subtraction (BGS) trained on *DS_complete* and trained using visible falls of *DS_restricted* but validated using *DS_complete*. Results from [28] using slightly different data set added as reference.

Method	SENS	PPV	TP	FN	FP
Robust BGS (using tracker and detector) for $\beta = 10$	45.5%	1.4%	10	12	688
Robust BGS only trained on visible falls for $\beta = 10$	40.9%	1.6%	9	13	526
Original BGS using APM [28] for $\beta = 10$	23.8%	2.2%	5	16	225
Original BGS using APM [28] for $\beta = 20$	38.1%	0.58%	8	13	1360

Table 3: Results of fall detection algorithm trained and validated using *DS_restricted*. Results of the same falls trained and validated on *DS_complete* added for comparison.

Person	Data	*DS_restricted*				Same falls from *DS_complete*			
		TP	FN	FP	FP/day	TP	FN	FP	FP/day
A	124.7 h	7	0	58	11.2	7	0	73	14
B	24 h	0	1	9	9	0	1	25	25
C	48 h	1	1	84	42	2	0	148	74
D	24 h	1	0	201	201	0	1	284	284
Total	220.7 h	9	2	352	38.3	9	2	530	57.6

detector that is trained using *DS_complete* show that still a high number of false alarms occurred. Only 10 out of 22 falls were detected, while 688 false alarms were generated. This represented a sensitivity of 45.5% and a PPV of 1.4%. Figure 5 shows a precision-recall (PR) curve of these results.

4.2. Training Using Visible Falls. As mentioned above, we also wanted to show the effect of using a more restricted data set for training the detector. In this case the person should be visible during the fall and he should remain on the floor for more than thirty seconds. The prior restriction made certain that the extracted features were more likely to be from the person and not from something else. Table 3 shows the results for each person using *DS_restricted* to validate the system. We also added the results of the same falls of *DS_restricted* but having been trained using *DS_complete* for comparison. This shows the effect of using the restricted training set in more depth.

The results using the more restricted training set show a large decrease of false alarms. Now nine out of eleven falls were detected while generating 352 false alarms, giving a sensitivity of 81.2% and a PPV of 2.49%. Looking to the amount of false alarms generated per day, a decrease of 33.5% from 57.6 to 38.3 can be observed.

However, two falls remained undetected of *DS_restricted*. The first was a fall in which person B lies down after sitting on the knees for 30 seconds while trying to get up again. The speed towards the ground was rather low. Additionally, a high level of overillumination was present in the image, interfering with a robust detection. In the second fall, the table is pushed away and person C is partially occluded during part of the fall. Figure 6 shows some screenshots of both missed falls.

From the 352 false detection instances, 21% was generated because a person was walking below the camera and was not completely visible anymore because of this. Another 26% was generated due to the presence of two persons or other moving objects in the room. In 13% of the cases, the person's movement was misclassified; this often happened together with errors caused by the background update. A partial occlusion of the body (e.g., the legs that were occluded by a table) also caused 10% of the errors; 9% of the false alarms were generated due to errors in the background update. In this case, the person was starting to be integrated in the background, or sometimes he was moving after being integrated in the background, which caused a ghost figure. Shadows (certainly the ones generated by the sun) also are still a challenge, since these accounted for 8% of the alarms. The other false alarms were caused by illumination changes, leaving or entering the view of the camera, and so forth.

Table 2 shows the results when training a model using *DS_restricted* and validating it on *DS_complete*. As mentioned in Section 3.3, tenfold cross-validation was used for the validation of the falls contained in *DS_restricted*. The other falls of *DS_complete* were classified using a model that was trained using *DS_restricted*. Table 4 shows these results more in detail over the different persons. This table also shows that the number of false alarms per day decreased from an average of 31.4 to 26 false alarms per day. These results are also contained in the PR curve shown in Figure 5. The doubling of the area under the curve (AUC) of the PR curve also shows the definite improvement in performance.

4.3. Personalized Fall Detector. To show the possibility that using personalization can give a further improvement, we have to look at the distribution of the false alarms per day over the different persons in Table 3. This shows that person D had a very high false alarm rate. In this case, 201 out of the total 352 false alarms were generated in a single video fragment of 24 hours. *DS_restricted* only contained one video of this person, so no data for person D was present in the training set. Person C also had a high false alarm rate of 42 alarms per day. But in this case two videos were present in the data set, one of them was available in the training set of the other one.

Table 4: Results per person for all videos from *DS_complete* using a model trained on *DS_restricted* and one trained using *DS_complete*.

Person	Data	Trained using *DS_restricted*				Trained using *DS_complete*			
		TP	FN	FP	FP/day	TP	FN	FP	FP/day
A	244.7 h	7	5	112	11.0	8	4	108	10.6
B	24 h	0	1	9	9	0	1	25	25
C	144 h	1	5	149	24.8	2	4	213	35.5
D	24 h	1	0	201	201	0	1	284	284
E	48 h	0	2	55	27.5	0	2	58	29
Total	484.7 h	9	13	526	26	10	12	688	31.4

Table 5: Results for personalization using online training for persons C and D for $Z = 4$. Different combinations of adding zero to five days are tested and averaged.

# videos	TP	FP	SENS (%)	PPV (%)
Person C				
0	0	176	0	0
1	0 (±0)	188.8 (±10.3)	0 (±0)	0 (±0)
2	0.2 (±0.42)	193.9 (±7.7)	3.3 (±7.2)	0.1 (±0.22)
3	0.8 (±0.42)	195.7 (±7.3)	13.3 (±7.2)	0.41 (±0.22)
4	1 (±0)	194.4 (±5.7)	16.7	0.51 (±0.15)
5	1	195	16.7	0.51
Person D				
0	1	201	100	0.49
1	0.8 (±0.45)	52 (±11.4)	80 (±44.7)	1.43 (±0.82)
2	0.9 (±0.32)	37.6 (±6.6)	90 (±31.6)	2.34 (±0.93)
3	1 (±0)	31.9 (±3.2)	100 (±0)	3.07 (±0.29)
4	1 (±0)	29.8 (±1.8)	100 (±0)	3.25 (±0.18)
5	1	29	100	3.33

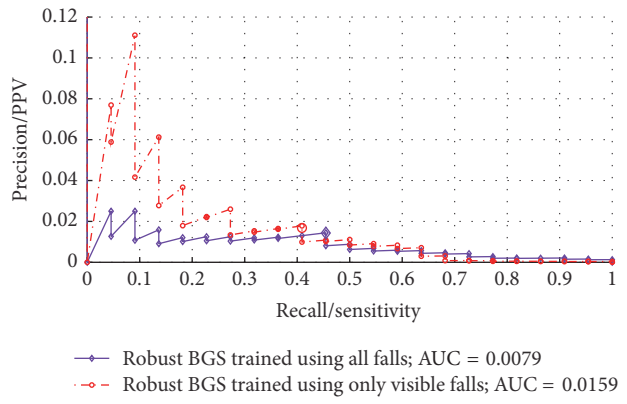


Figure 5: Precision-recall curve for comparison of our robust background subtraction (BGS) algorithm trained on *DS_complete* containing all falls and trained only using *DS_restricted* containing only visible falls but validated on *DS_complete*. Both use $\beta = 10$ for the F_β-measure. The larger markings indicate the optimal trained model (AUC = Area Under Curve).

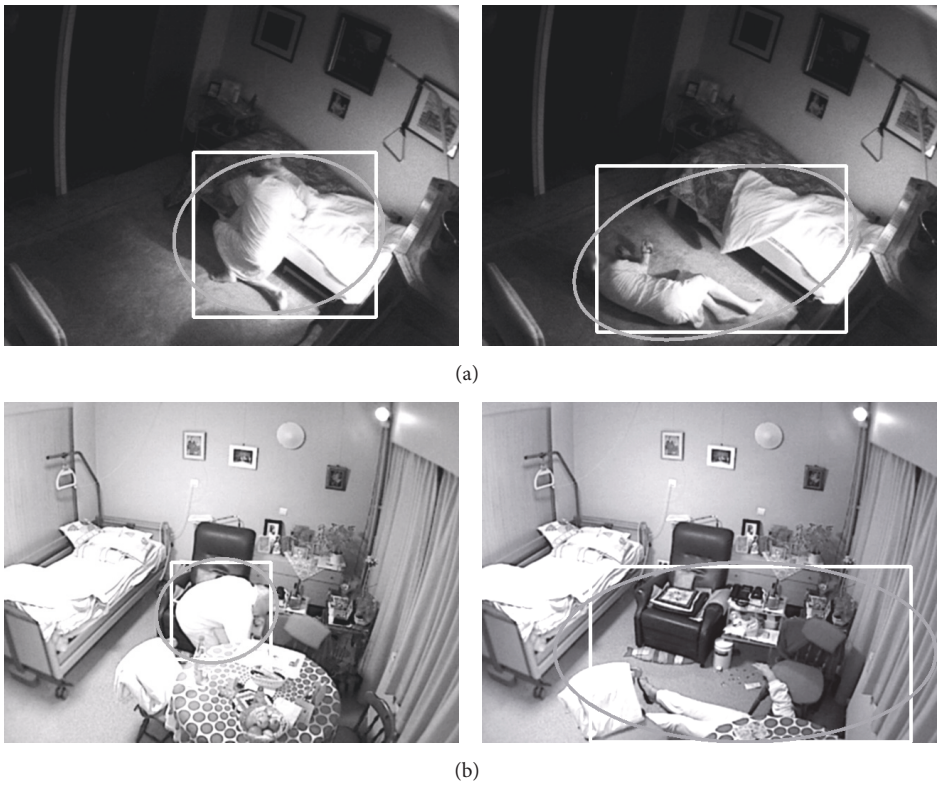
This could be the reason that person C has a lower amount of false alarms than person D.

To reduce the number of false alarms further, two additional experiments, one for person C and one for person D, to show the effects of personalization using online learning were executed. For this, all fall videos from *DS_restricted* excluding the ones from the current person were combined with additional videos from this person from *DS_personalized* containing only normal activities. Several combinations and number of additional videos were tested. A fall detector was trained for every data set. Different values for Z to alter the weight parameter of the SVM were tested. Persons C and D were used independently for cross-validation to find an optimal value for Z. A value of four gave the best results in both cases. The results for $Z = 4$ are shown in Table 5.

The results show a different behavior for both persons. The results for person D showed a large decrease of the false positives from 201 to 29 while keeping the sensitivity. For person C, a small increase of the false alarms was noticed, but more importantly also the sensitivity increased. Remember that in the previous experiments there always was a video containing a fall of person C in the training set of the other ones; this explains the higher number of false alarms as found in Table 3. An overview of the causes of false alarms per person when using five additional days is shown in Table 6.

5. Discussion

5.1. Robust Foreground Detection Validated Using DS_complete. The tests using our complete data set *DS_complete* of twenty-two real-life videos showed an improvement of



(a)

(b)

Figure 6: Screenshots of both falls of *DS_restricted* that were not detected. (a) Person B lies down after sitting on the knees. (b) Person C pushes the table away while falling; only the legs are visible during and directly after the fall.

Table 6: Causes of false positives when using online training with five additional days.

Error cause	Person C	Person D
Walk under camera	24%	35%
Other moving objects	25%	14%
Other persons	18%	14%
Legs occluded	16%	21%
Illumination change	4%	0%
Ghost formation	7%	3%
Others	6%	13%

the results by using a particle filter in combination with a person detector to create more robust foreground detection. Comparing the results with previous experiments using a less robust BGS shows almost a decrease of the amount of false alarms by 50% while retaining or even increasing the sensitivity. This shows that using a robust foreground segmentation is important for camera-based fall detection algorithms. This reduction is certainly substantial, but still a high amount of false alarms were generated, while failing to detect twelve out of twenty-two falls.

5.2. Training Using Visible Falls. Looking into more details to the different available falls showed that a high number of the falls were partially or even completely occluded. Only in 50% of the falls was the person visible after the fall. This caused problems since our foreground detection was not written to be occlusion resistant. When the person was not visible during or after the fall, the features that were extracted from the foreground region were not representative for the person. Hence, training a fall detector using these erroneous features values caused it to be less robust. Removing these falls from the data set in *DS_restricted* and training our fall detector using this subset showed an improvement. Most importantly, the AUC of the PR curve of the fall detector (see Figure 5) trained using only visible falls doubled opposed to using all falls. The false alarms rate decreased from 31.4 to 26 falls/day. The sensitivity also decreased from 45.5% to 40.9%. One fall less was detected. However this fall was one in which the person was not visible, so it could be lucky detection.

The other falls could not be detected by our current non-occlusion-resistant algorithm. This shows that it is important to be able to cope with occlusions. This could partially be done by using only head tracking and detecting occlusions using this, or by placing more cameras in the room. It also shows that it is important to only use features that were extracted correctly and not to include noisy falls in the training data. As discussed in Section 4, the two falls from *DS_restricted* that were not detected were very challenging. The missed fall from person C was also almost completely occluded during the fall, but the person was visible shortly

after the fall. Maybe it should have been removed from the restricted data set. Still, even when using only nonoccluded falls to train the model of the fall detector quite a high number of false alarms were generated.

5.3. Personalized Fall Detector. Each person has his own way of living and walking. Training a fall detector that is able to detect all falls and generate as few false alarms as possible is still very challenging. Analyzing the false alarms of person D showed that her walking speed was much higher than the other participants. Also the speed while bending over to take something out of the cupboard seemed a lot higher. Additional to this, the setup of the room and the camera view differed a lot from the others. Our belief is that it is better to use a generic detector as a base and let it learn the specific patterns of the person during the usage of the system. Our results supported this with showing an improvement in the robustness of the system. A large decrease of the false alarm rate was noticed for person D, while an increase of the sensitivity with only a minor increase of the false alarms was produced for person C. It would be better to test this on more persons, but unfortunately no additional persons with visible falls were available in our real-life data set. Adding more days would probably further decrease the number of generated false alarms, but only up to a certain point. To decrease the number of false alarms further, it is important to solve the challenges that cause errors in the foreground detection.

A personalized detector provides the possibility of integrating more environmental information in the future. One of the main error causes, for example, is that the person was walking below the camera. Border object detection could prevent these errors. A further option for personalization is to train the person detector to learn the appearance of the monitored person. This would improve the performance of the foreground detection and also could provide a means to detect occlusions.

5.4. Future Work. As mentioned before, the performance of this kind of fall detection algorithm depends heavily on the placement of the camera in the room. To cover the room more cameras are needed, but even then still dead spots remain possible. To increase the robustness further, our belief is that it is best to integrate two different ways of detecting a fall. The system proposed in this paper based on transient information could be combined with a detector that is based on contextual information. These kinds of systems first need to learn the normal pattern of the person and then can detect deviations from this. Combining this kind of information could improve the robustness of the system a lot. In addition to combining different algorithms using the same sensor, also improvement in performance can be found in combining several types of sensors. This has currently received a lot of attention, but it has its own challenges and issues as described in [51].

6. Conclusion

In this paper, we showed three contributions to enhance the performance of our fall detector. As a first contribution, this paper showed the results of our new approach using our real-life data set. The use of a more robust foreground detection by integrating person detection and tracking to segment the person from the background increases the performance of our fall detection algorithm. It reduced the number of false alarms with 50% compared to our previous system while maintaining or even improving the sensitivity. However, the numerous occluded falls that were included in our real-life fall data set still caused the detector to generate a high number of false alarms. As a second contribution, we showed that only using nonoccluded falls in the training set reduced the false alarms from 31.4 to 26 per day. The AUC of this detector using selected falls was twice as large as the detector using a model trained using all available falls. But we saw that some persons have a higher number of false alarms than others. This could partially be explained by having less training data available for these persons. As a third contribution, we showed that personalization of the model used for the classification of falls can further improve the performance. Our approach of online training using only ADL from the person itself in addition to a generic data set further increased the robustness of our camera-based fall detection algorithm by reducing the false alarm rate by a factor 7 in one case, while increasing the sensitivity of the system with 17% for an increase of the false alarms of 11%. These optimizations provide a step forward in solving the fall detection challenge, but even then adding other cameras or sensors may be needed for a practical real-life system.

Appendix

A. Foreground Segmentation

Our foreground segmentation is based on an approximate median (APM) filter (see Figures 2(a)–2(c)) combined with a particle filter (PF) to increase its robustness. Our implementation of this PF uses a combination of foreground segmentation, a weighted structural intensity histogram, and a person detector to follow the person in the image. To make the background update more robust, the prediction of the PF was used as feedback. The BG was updated more slowly inside of the prediction of the PF and faster outside of this region. This reduced the formation of a ghost figure while other changes in the image (e.g., changes in lighting, other moving objects) were integrated faster. The different parts of our foreground segmentation are explained in more detail below.

A.1. Particle Filter. Particle filters estimate the probability distribution $p(S_t \mid Z[1 \cdots t])$ of the state vector S_t of the tracked object given Z_t representing all the observations. This probability density function can be approximated using a set of N weighted samples or particles. Increasing N makes the particle filter more robust, but also the time to process a single frame increases accordingly. Initial tests showed us that using 80 particles gave a good trade-off between processing time and accuracy.

Each particle corresponds with one state vector representing an ellipse that is defined by its center coordinates, major and minor axis length, and angle of the major axis and the ground plane. Also the speed at which each of these values changes is recorded as part of the state vector, resulting in a ten-dimensional state vector. The weight of each particle is updated every frame and depends on the previous state and the current measurement function. Our measurements are based on foreground, weighted structural histogram, and person detection coefficients. These are explained in more detail below. The used particle filter is a Bootstrap filter implemented using the Bayesian Filtering Library [52]. To start tracking, an initialization step is needed. In our case, the tracker was initialized when a foreground object of over 5000 pixels was detected in our image of 640 by 480 pixels.

A.2. Foreground Coefficient C_F. The foreground coefficient (see Figure 2(f)) measures how well the ellipse fits the foreground object. The foreground was detected by thresholding the difference between the current frame and the background. If the difference in intensity level was more than eight, the pixel was detected as foreground. Unfortunately, also shadows could be included in the foreground this way. These were removed using cross correlation. This was followed by an erosion/dilation step to remove small noisy patches.

A high value for the foreground coefficient represented an ellipse that surrounds the foreground as well as possible without including too many background pixels. As shown on Figure 2(f), two layers were defined surrounding this bounding ellipse (BE). A penalty was given if foreground pixels were included in either of these layers. The exact value of C_F was calculated with the following formula:

$$C_F = 1.2\frac{\mathrm{FG}_{\mathrm{BE}}}{Z_{\mathrm{BE}}} - 0.6\frac{\mathrm{FG}_{\mathrm{OL}}}{Z_{\mathrm{OL}}} - 0.4\frac{\mathrm{FG}_{\mathrm{OOL}}}{Z_{\mathrm{OOL}}}, \quad (A.1)$$

where $\mathrm{FG}_{\mathrm{BE}}$ is the amount of foreground pixels contained in the bounding ellipse (BE), Z_{BE} is the surface of BE, OL is a layer surrounding BE 1.5 times the size of BE, and OOL is an additional layer twice the size of BE.

A.3. Weighted Structural Histogram Coefficient C_H. Histogram matching of the image in the bounding ellipse around the person is the base for our second measurement function (see Figure 2(e)). In the literature, mostly a color histogram is used because of it being more robust. But since also video was recorded during the night using near-infrared, only gray-scale values are available. A weighted structural histogram was used to make the histogram more distinctive [53]. The bounding ellipse was divided into four overlapping circles as seen in Figure 2(e). Each circle represented a different part of the body, more precisely the head and shoulders, the chest, the abdomen and hips, and the lower legs. Since it is more probable that some background pixels are included in this circle at the edges, the center pixels were given more weight than the ones on the outside of the circle using a Gaussian distribution. One exception to this rule was the circle containing the legs. When a person is walking, his legs move all over the circle causing the center to contain background information at some points. Therefore the weights were evenly distributed over this whole circle. In the literature the pixels are mostly included in only one bin of the histogram. Even small intensity changes can cause a pixel to shift to another bin, sometimes causing dramatic changes. To reduce this effect, linear interpolation was used to divide the weight of the pixels over the bin and its neighbors.

Calculating the correspondence of two histograms can be done in different ways, like Bhattacharyya or chi-squared distance, but our tests showed that correlation with the histogram model (H_M) gave the best results. To calculate C_H, the correlation of the histograms for each part of the ellipse was calculated and combined as given in

$$C_H = 0.3C_{\mathrm{head}} + 0.35C_{\mathrm{chest}} + 0.25C_{\mathrm{abdomen}} + 0.1C_{\mathrm{legs}}. \quad (A.2)$$

During the initialization the histogram was calculated from the biggest object and used as the starting model. The appearance of the person changes while moving, which could cause the tracker to lose track. To counteract this, H_M was updated during each frame with 0.5% of the current prediction. This was done by multiplying all bins of H_M with 0.995 and adding the histogram of the current prediction multiplied with 0.005 to this.

A.4. Person Detection Coefficient C_D. Not only persons move through the room. Also other objects, such as walking aids, can be detected as foreground. To reduce the effect of these other objects, the Calvin upper-body detector [46] was used (see Figure 2(g)). This detector is based on the successful part-based object detection framework [54]. Unfortunately, it takes a huge amount of labeled training data to train a new model, so a standard model was used. We chose an upper-body detector because, in contrast to most pedestrian detectors, it can also detect sitting persons. Another major advantage of the Calvin detector is that the model is available online and can easily be used in the OpenCV framework. The detector returns a confidence level for each detection instance that it makes. Our tests showed that a value of over −0.45 represents a high confidence. But, as explained before, the used model is not trained specifically for our data. The higher point of view that was used differs from the data on which the model was trained. Also the posture of older persons, or the presence of a lump on their back, differs from younger people. It can only detect upper bodies in an upright position, so persons that are lying down can not be detected anymore. This caused quite some false and missed detection instances, but, even with these drawbacks, the detector proved useful.

The detection was used in three ways: if a person was detected, H_M was updated in the same way as described above, but with a multiplication coefficient depending on the confidence of the detection. When the detection is very good, the histogram model is completely replaced with the histogram of this detected person. If the confidence level is lower, the value with which H_M was multiplied was defined using a Gaussian centered around −0.45 with $\sigma^2 = 0.15$. If one or more person detection instances were available, also five low weight particles were replaced by a combination of these

detection instances. The replaced particles were unlikely to correspond to the person but are now directly placed on this detected person. This is normally not done in the particle filter paradigm, but it increased the robustness of the tracker. To limit the processing time the detection was only executed every five frames on a region that is centered around the detected foreground object but is 80% bigger. Additionally, also the calculation of the final coefficient is changed. How is explained next.

A.5. Final Coefficient. Finally, the final coefficient was calculated as a combination of the foreground C_F and histogram coefficient C_H. The detection coefficient C_D was used to shift the weight between these both measures. The formula for the final coefficient was given by

$$C_{\text{total}} = wC_F + (1 - w)\, C_H. \tag{A.3}$$

During the initialization step, w was set to 0.65. When a detection instance was available, w was decreased according to the confidence of the detection as used for the update of the histogram model. For a reliable detection, w is decreased to 0.25. This increases the importance of C_H and decreases this of C_H. Values for w lower than 0.25 were clipped to make certain that the tracker does not stick to a nonmoving object. When no detection was available, w was gradually increased to 0.65 again. This reduces the chance that another object is tracked. If no detection was available for 20 frames, the initial value for w was used again.

A.6. Predicted State of Particle Filter. As mentioned above, a particle filter represents a probability density function using particles. From this function, a prediction can be calculated (see Figure 2(h)). In our case, the mean of the five best predictions was used. This prediction was found to be more stable than the weighted mean of all particles or the particle with the highest weight. This predicted ellipse was also used as feedback for the update of the background. To make the background update more robust, the background was updated more slowly inside of the prediction of the PF and faster outside of this region.

A.7. Clean-Up. To reduce the impact of erroneous foreground detection due to, for example, the continuous update of the background or spots from the sun, small blobs were omitted. The same size as for the initialization of the PF was used.

A.8. Processing Time. Unfortunately the usage of this kind of robust tracker has an effect on the processing time. We used a PC with 16 GB of RAM and an Intel Core i5-4670 CPU running on 3.40 GHz from 2013. The current processing time of one frame depends on the size of the person in the image, but it is situated between 250 and 600 ms when no person detection is executed. As stated above, every five frames, the upper-body detector is executed using part of the image. This adds between 100 and 300 ms to the processing time of that frame. Almost 90% of the processing time is needed to calculate the coefficient of all particles. This takes between 1 and 8 ms per particle multiplied by the number of particles, 80 in our case. The processing speed can be increased by optimizing and reimplementing some parts of the code. But the highest performance gain can be expected by parallelizing the calculation of the coefficients of the particles and executing this using the GPU of the PC.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is funded by the "Fonds voor Wetenschappelijk Onderzoek" (FWO) via Project G039811N: "Multi-Camera Human Behavior Monitoring and Unusual Event Detection," by the "Agentschap Innoveren en Ondernemen" Vlaio via "Technology Transfer" (TETRA) Project 80150 "FallCam: Detection of Fall in Older Persons with a Camera System," and by the EU via EraSME (FP7) Project (IWT) 100404 "AMACS: Automatic Monitoring of Activities using Contactless Sensors." It is also funded by the Interdisciplinary Institute for Technology (iMinds) FallRisk Project. The iMinds FallRisk project is cofunded by iMinds, a research institute founded by the Flemish Government. Companies and organizations involved in the project are COMmeto, Televic Healthcare, TP Vision, Verhaert, and Wit-Gele Kruis Limburg, with project support of IWT. The authors thank the persons who participated in the research by giving their permission to be monitored over several months. The authors also thank all of their colleagues of the FallCam and AMACS project who helped in acquiring the real-life data set. The authors would also like to acknowledge the Prevention of Falls Network for Dissemination (ProFouND) and the Cooperation in Science and Technology (COST) Algorithms, Architectures and Platforms for Enhanced Living Environments (AAPELE) network.

References

[1] K. Milisen, E. Detroch, K. Bellens et al., "Falls among community-dwelling elderly: a pilot study of prevalence, circumstances and consequences in Flanders," *Tijdschrift voor Gerontologie en Geriatrie*, vol. 35, no. 1, pp. 15–20, 2004.

[2] J. Fleming and C. Brayne, "Inability to get up after falling, subsequent time on floor, and summoning help: prospective cohort study in people over 90," *British Medical Journal*, vol. 337, no. 7681, pp. 1279–1282, 2008.

[3] J. L. Redd, R. D. Zura, A. E. Tanner, E. E. Walk, M. M. Wu, and R. F. Edlich, "Personal emergency response systems," *Journal of Burn Care & Rehabilitation*, vol. 13, no. 4, pp. 453–459, 1992.

[4] N. Noury, A. Fleury, P. Rumeau et al., "Fall detection—principles and methods," in *Proceedings of the 29th Annual International Conference of IEEE-EMBS (Engineering in Medicine and Biology Society (EMBS '07)*, pp. 1663–1666, August 2007.

[5] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *Proceedings of the 2008 10th International*

Conference on e-Health Networking, Applications and Service, (HEALTHCOM '08), pp. 42–47, IEEE, Singapore, Singapore, July 2008.

[6] J. T. Perry, S. Kellog, S. M. Vaidya, J.-H. Youn, H. Ali, and H. Sharif, "Survey and evaluation of real-time fall detection approaches," in *Proceedings of the 6th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET '09)*, pp. 158–164, IEEE, Alexandria, Egypt, December 2009.

[7] F. Hijaz, N. Afzal, T. Ahmad, and O. Hasan, "Survey of fall detection and daily activity monitoring techniques," in *Proceedings of the 2nd International Conference on Information and Emerging Technologies (ICIET '10)*, pp. 1–6, IEEE, Karachi, Pakistan, Pakistan, June 2010.

[8] F. Cardinaux, D. Bhowmik, C. Abhayaratne, and M. S. Hawley, "Video based technology for ambient assisted living: a review of the literature," *Journal of Ambient Intelligence and Smart Environments*, vol. 3, no. 3, pp. 253–269, 2011.

[9] G. Ward, N. Holliday, S. Fielden, and S. Williams, "Fall detectors: a review of the literature," *Journal of Assistive Technologies*, vol. 6, no. 3, pp. 202–215, 2012.

[10] G. Yongli, O. Shih Yin, and P. Y. Han, "State of the art: a study on fall detection , World Academy Science," *Engineering and Technology*, vol. 62, pp. 294–298, 2012.

[11] S. Chaudhuri, H. Thompson, and G. Demiris, "Fall detection devices and their use with older adults: a systematic review," *Journal of Geriatric Physical Therapy*, vol. 37, no. 4, pp. 178–196, 2014.

[12] M. A. Habib, M. S. Mohktar, S. B. Kamaruzzaman, K. S. Lim, T. M. Pin, and F. Ibrahim, "Smartphone-based solutions for fall detection and prevention: challenges and open issues," *Sensors*, vol. 14, no. 4, pp. 7181–7208, 2014.

[13] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical Engineering Online*, vol. 12, 2013.

[14] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[15] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, "Automatic fall monitoring: a review," *Sensors*, vol. 14, no. 7, pp. 12900–12936, 2014.

[16] S. S. Khan and J. Hoey, "Review of fall detection techniques: a data availability perspective," *Medical Engineering & Physics*, vol. 39, pp. 12–22, 2017.

[17] F. J. S. Thilo, B. Hürlimann, S. Hahn, S. Bilger, J. M. G. A. Schols, and R. J. G. Halfens, "Involvement of older people in the development of fall detection systems: a scoping review," *BMC Geriatrics*, vol. 16, no. 1, article no. 42, 2016.

[18] Z. Zhang, C. Conly, and V. Athitsos, "A survey on vision-based fall detection," in *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments, (PETRA '15)*, ACM, Corfu, Greece, July 2015.

[19] Y. Nizam, M. N. Mohd, and M. M. Jamil, "A study on human fall detection systems: Daily activity classification and sensing techniques," *International Journal of Integrated Engineering*, vol. 8, no. 1, 2016.

[20] SeniorWatch, "Fall detector: Case study of european ist seniorwatch project," Tech. Rep., SeniorWatch, 2001.

[21] L. Alhimale, H. Zedan, and A. Al-Bayatti, "The implementation of an intelligent and video-based fall detection system using a neural network," *Applied Soft Computing*, vol. 18, pp. 56–69, 2014.

[22] M. Belshaw, B. Taati, S. Jasper, and A. Mihailidis, "Towards a single sensor passive solution for automated fall detection," in *Proceeding of the Annual International Conference on Engineering in Medicine and Biology Society, (EMBC '11)*, pp. 1773–1776, IEEE, Boston, Mass, USA, 2011.

[23] P. Feng, M. Yu, S. M. Naqvi, and J. A. Chambers, "Deep learning for posture analysis in fall detection," in *Proceedings of the 2014 19th International Conference on Digital Signal Processing, (DSP '14)*, pp. 12–17, Hong Kong, China, August 2014.

[24] H. Foroughi, B. S. Aski, and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," in *Proceedings of the 11th International Conference on Computer and Information Technology, (ICCIT '08)*, pp. 219–224, IEEE, Khulna, Bangladesh, December 2008.

[25] M. Krekovic, P. Ceric, T. Dominko et al., "A method for real-time detection of human fall from video," in *Proceedings of the 35th International Convention (MIPRO '12)*, pp. 1709–1712, IEEE, Opatija, Croatia, 2012.

[26] M. Yu, Y. Yu, A. Rhuma, S. M. R. Naqvi, L. Wang, and J. A. Chambers, "An online one class support vector machine-based person-specific fall detection system for monitoring an elderly individual in a room environment," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 6, pp. 1002–1014, 2013.

[27] G. Debard, P. Karsmakers, M. Deschodt et al., "Camera-based fall detection on real world data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7474, pp. 356–375, 2012.

[28] G. Debard, M. Mertens, M. Deschodt et al., "Camera-based fall detection using real-world versus simulated data: How far are we from the solution?" *Journal of Ambient Intelligence and Smart Environments*, vol. 8, no. 2, pp. 149–168, 2016.

[29] G. Debard, G. Baldewijns, T. Goedemé, T. Tuytelaars, and B. Vanrumste, "Camera-based fall detection using a particle filter," in *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, (EMBC '15)*, pp. 6947–6950, IEEE, Milan, Italy, August 2015.

[30] G. Baldewijns, G. Debard, G. Mertes, B. Vanrumste, and T. Croonenborghs, "Bridging the gap between real-life data and simulated data by providing a highly realistic fall dataset for evaluating camera-based fall detection algorithms," *Healthcare technology letters*, vol. 3, no. 1, pp. 6–11, 2016.

[31] C. Medrano, I. Plaza, R. Igual, Á. Sánchez, and M. Castro, "The effect of personalization on smartphone-based fall detectors," *Sensors*, vol. 16, no. 1, article no. 117, 2016.

[32] E. Auvinet, F. Multon, A. Saint-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-D silhouette vertical distribution," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 2, pp. 290–300, 2011.

[33] S. Zambanini, J. Machajdik, and M. Kampel, "Early versus late fusion in a multiple camera network for fall detection," in *Proceedings of the 4th Annual Workshop of the Austrian Association f. Pattern Recognition*, pp. 15–22, Zwettl, Austria.

[34] X. Ma, H. Wang, B. Xue, M. Zhou, B. Ji, and Y. Li, "Depth-based human fall detection via shape features and improved extreme learning machine," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 6, pp. 1915–1922, 2014.

[35] E. E. Stone and M. Skubic, "Fall detection in homes of older adults using the microsoft kinect," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 1, pp. 290–301, 2015.

[36] Y. Chen, C. Chuang, C. Yu, and K. Fan, "Fall detection in dusky environment," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, vol. 260 of *Lecture Notes in Electrical Engineering*, pp. 1131–1138, Springer, Berlin, Germany, 2014.

[37] M. Yu, A. Rhuma, S. M. Naqvi, L. Wang, and J. Chambers, "A posture recognition-based fall detection system for monitoring an elderly person in a smart home environment," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1274–1286, 2012.

[38] W.-Y. Shieh and J.-C. Huang, "Falling-incident detection and throughput enhancement in a multi-camera video-surveillance system," *Medical Engineering & Physics*, vol. 34, no. 7, pp. 954–963, 2012.

[39] W. Feng, R. Liu, and M. Zhu, "Fall detection for elderly person care in a vision-based home surveillance environment using a monocular camera," *Signal, Image and Video Processing*, vol. 8, no. 6, pp. 1129–1138, 2014.

[40] S. Albawendi, K. Appiah, H. Powell, and A. Lotfi, "Video based fall detection with enhanced motion history images," in *Proceedings of the 9th ACM International Conference on Pervasive Technologies Related to Assistive Environments, (PETRA '16)*, ACM, Corfu, Island, Greece, July 2016.

[41] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, vol. 8, no. 3, pp. 187–193, 1995.

[42] G. Bradski, ""Opencv," Dr. Dobb's Journal of Software Tools, 2000".

[43] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, pp. 28–31, IEEE, Cambridge, UK, August 2004.

[44] A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," in *Proceedings of the 2012 American Control Conference, (ACC '12)*, pp. 4305–4312, IEEE, Montreal, QC, Canada, June 2012.

[45] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[46] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari, "2D articulated human pose estimation and retrieval in (almost) unconstrained still images," *International Journal of Computer Vision*, vol. 99, no. 2, pp. 190–214, 2012.

[47] J. Nino-Castaneda, A. Frias-Velaqzquez, N. Bo Bo et al., "Scalable semi-automatic annotation for multi-camera person tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2259–2274, May 2016.

[48] L. Young-Sook and L. HoonJae, "Multiple object tracking for fall detection in real-time surveillance system," in *Proceeding of the 11th International Conference on Advanced Communication Technology, (ICACT '09)*, vol. 3, pp. 2308–2312, IEEE, Phoenix Park, South Korea, 2009.

[49] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Fall detection from human shape and motion history using video surveillance," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia, (AINAW '07)*, pp. 875–880, IEEE, Niagara Falls, Ont., Canada, May 2007.

[50] N. Noury, P. Rumeau, A. K. Bourke, G. ÓLaighin, and J. E. Lundy, "A proposal for the classification and evaluation of fall detectors," *IRBM*, vol. 29, no. 6, pp. 340–349, 2008.

[51] G. Koshmak, A. Loutfi, and M. Linden, "Challenges and issues in multisensor fusion approach for fall detection: review paper," *Journal of Sensors*, vol. 2016, Article ID 6931789, 12 pages, 2016.

[52] K. Gadeyne, "BFL: bayesian filtering library," http://www.orocos.org/bfl

[53] A. Jacquot, P. Sturm, and O. Ruch, "Adaptive tracking of non-rigid objects based on color histograms and automatic parameter selection," in *Proceedings of the IEEE Workshop on Motion and Video Computing, (MOTION '05)*, vol. 2, pp. 103–109, IEEE, Breckenridge, CO, USA, January 2005.

[54] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, Anchorage, Alaska, June 2008.